



MANUAL TÉCNICO

USAC PROCESSING
DATA

IPC1

CARNET 202407596

NOMBRE: Jefferson Ajcúc

I. INTRODUCCION

La práctica "USAC Processing Data" se desarrolla en Java utilizando el patrón MVC. El sistema permite cargar un archivo CSV, visualizar los datos en un gráfico de barras, aplicar diferentes algoritmos de ordenamiento de manera interactiva y generar reportes en PDF. Para ello, se emplean Swing, JFreeChart e iText, además de hilos para mostrar en tiempo real el proceso de ordenamiento, incluyendo el tiempo transcurrido y la cantidad de pasos realizados. Este manual técnico describe la arquitectura del sistema, las funcionalidades implementadas y las decisiones de diseño adoptadas, facilitando su comprensión, mantenimiento y extensión futura.

II. OBJETIVOS

Generales ● Familiarizar al estudiante con el lenguaje de programación JAVA. ● Que el estudiante aplique los conocimientos adquiridos en el curso de Introducción a la Programación y computación 1. ● Elaborar la lógica para presentar una solución a la propuesta planteada. Específicos ● Uso del lenguaje de programación Java como herramienta de desarrollo de software. ● Construcción de aplicaciones simples con interfaz gráfica. ● Implementación de sentencias de control, ciclos y vectores. ● Manejo de los algoritmos de ordenamiento. ● Implementación de hilos para la realización de múltiples tareas en paralelo.

III. Dirigido

Este manual está orientado a desarrolladores de software, ingenieros de sistemas, y profesionales de TI interesados en entender a fondo el diseño y la lógica detrás del sistema. Está diseñado para aquellos que buscan mantener, ampliar o adaptar el sistema, así como para aquellos interesados en aprender mas sobre como funciona el sistema detras de el programa usac international bank

IV. Especificación Técnica

Requisitos de Hardware:

- Computadora de escritorio o portátil.
- Al menos 4 GB de memoria RAM.
- Disco duro con al menos 250 GB de capacidad.
- Procesador Intel Core i3 o superior.
- Resolución gráfica mínima de 1024 x 768 píxeles.

Requisitos de Software:

- Sistema Operativo: Windows 7 o superior.
- Java Runtime Environment (JRE): versión 8.2 o superior.
- Java Development Kit (JDK): versión 8.2 o superior.
 - Bibliotecas externas: iText (librería para generación de reportes PDF), versión 5.5.13 o superior.
- Entorno de desarrollo recomendado: Apache NetBeans IDE 8.2 o superior.

Bibliotecas internas requeridas:

- Biblioteca gráfica Swing (parte del JDK estándar).
- Biblioteca gráfica AWT (Abstract Window Toolkit, parte del JDK estándar).

Librerías externas requeridas:

- com.itextpdf (iText), versión 5.x o compatible.

IV. LÓGICA DEL PROGRAMA

Nombre del Paquete	Descripción del Paquete	Clases en el Paquete
USACProcessingData.Controller	Contiene la lógica de control y la comunicación entre la capa de vista y la capa de modelo. Gestiona la carga de datos y la manipulación principal.	MainController
USACProcessingData.Model	Representa la capa de datos y la lógica de negocio: maneja la carga de archivos CSV, la estructura de datos, los algoritmos de ordenamiento y la generación de reportes en PDF.	CSVLoader, DataEntry, Sorter, PDFReportGenerator
USACProcessingData.View	Incluye todas las ventanas y componentes gráficos de la aplicación, donde el usuario interactúa con el sistema.	MainFrame, SortDialog
USACProcessingData (raíz)	Contiene la clase principal que inicia la aplicación.	Main

CLASE: Main

Esta clase es la principal del programa y se encarga de inicializar y lanzar la aplicación.

Clase: Main

La clase Main es la principal del programa y se encarga de iniciar la ejecución de la aplicación. Desde aquí se invoca la interfaz gráfica principal (MainFrame), estableciendo el punto de entrada para que el usuario pueda interactuar con el sistema. No cuenta con atributos o variables propios; simplemente crea y muestra la ventana principal al arrancar.



```
package USACProcessingData;

import javax.swing.SwingUtilities;
import USACProcessingData.View.MainFrame; // Importar la clase MainFrame

public class Main {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new MainFrame().setVisible(true);
        });
    }
}
```

CLASE: MainController

Lla clase CreateAccountController es responsable de manejar la lógica para la creación de nuevas cuentas bancarias en el sistema. Coordina la interacción entre la interfaz gráfica, la validación de límites por cliente, la generación automática del identificador de la cuenta, y el registro de la operación en la bitácora.

ATRIBUTOS Y VARIABLES

Elemento	Descripción
Atributos	- dataList: Lista de objetos DataEntry con los datos cargados desde el CSV.
	- xLabel: Etiqueta para el eje X, obtenida del encabezado del archivo CSV.
	- yLabel: Etiqueta para el eje Y, obtenida del encabezado del archivo CSV.
Métodos	- loadData(File): Carga los datos del archivo CSV y actualiza dataList, xLabel y yLabel.
	- getDataList(): Retorna la lista de datos cargados.
	- getXLabel(): Retorna la etiqueta del eje X.
	- getYLabel(): Retorna la etiqueta del eje Y.

MÉTODO: loadData

Carga el archivo CSV utilizando CSVLoader, actualizando la lista de datos y las etiquetas de los ejes. Retorna true si la carga es exitosa, o false en caso de error.

```
public boolean loadData(File file) {
    try {
        CSVLoader loader = new CSVLoader();
        this.dataList = loader.loadFromFile(file);
        this.xLabel = loader.getXLabel();
        this.yLabel = loader.getYLabel();
        return true;
    } catch (IOException e) {
        e.printStackTrace();
        return false;
    }
}
```

MÉTODO: getDataList

- Retorna la lista de objetos DataEntry que contienen los datos leídos desde el archivo CSV.

```
public List<DataEntry> getDataList() {  
    return dataList;  
}
```

CLASE: SortDialog

SortDialog es una ventana modal que permite al usuario configurar y visualizar en tiempo real el proceso de ordenamiento de datos. La interfaz muestra las opciones para seleccionar el algoritmo, tipo de orden (ascendente o descendente) y velocidad, actualizando simultáneamente estadísticas como el tiempo transcurrido y el número de pasos realizados, y al finalizar genera un reporte PDF con toda la información del proceso.

ATRIBUTOS Y VARIABLES

Atributos	Variables
Lista de datos originales del CSV	dataList
Copia de los datos para realizar el ordenamiento	sortingData
Etiqueta para el eje X	xLabel
Etiqueta para el eje Y	yLabel
Componente para seleccionar el algoritmo de ordenamiento	algorithmComboBox
Botón para seleccionar orden ascendente	ascendingRadio
Botón para seleccionar orden descendente	descendingRadio
Componente para seleccionar la velocidad de ordenamiento	speedComboBox
Botón para iniciar el proceso de ordenamiento	startButton
Etiqueta que muestra el algoritmo seleccionado	algorithmLabel
Etiqueta que muestra la velocidad seleccionada	speedLabel
Etiqueta que indica el tipo de orden (ascendente o descendente)	orderTypeLabel
Etiqueta que muestra el tiempo transcurrido durante el ordenamiento	timeLabel
Etiqueta que muestra la cantidad de pasos realizados	stepsLabel
Panel donde se visualiza la gráfica	chartPanel
Objeto encargado de gestionar los algoritmos de ordenamiento	sorter

MÉTODO: initUI

Configura y organiza los componentes de la interfaz (opciones, estadísticas y panel gráfico).

```
private void initUI() {
    setLayout(new BorderLayout());

    // Panel con las opciones
    JPanel optionsPanel = new JPanel(new GridLayout(2, 1, 5, 5));
    JPanel topOptions = new JPanel(new FlowLayout());

    // Selección de algoritmo
    algorithmComboBox = new JComboBox<>(new String[]{
        "Bubble Sort", "Insert Sort", "Select Sort", "Merge Sort", "Quicksort", "Shellsort"
    });
    topOptions.add(new JLabel("Algoritmo:"));
    topOptions.add(algorithmComboBox);

    // Orden asc/desc
    ascendingRadio = new JRadioButton("Ascendente", true);
    descendingRadio = new JRadioButton("Descendente");
    ButtonGroup orderGroup = new ButtonGroup();
    orderGroup.add(ascendingRadio);
    orderGroup.add(descendingRadio);
    topOptions.add(new JLabel("Tipo:"));
    topOptions.add(ascendingRadio);
    topOptions.add(descendingRadio);

    // Selección de velocidad
    speedComboBox = new JComboBox<>(new String[]{"Alta", "Media", "Baja"});
    topOptions.add(new JLabel("Velocidad:"));
    topOptions.add(speedComboBox);

    // Botón iniciar ordenamiento
    startButton = new JButton("Iniciar Ordenamiento");
    topOptions.add(startButton);

    optionsPanel.add(topOptions);

    // Panel de estadísticas
    JPanel statsPanel = new JPanel(new FlowLayout());
    algorithmLabel = new JLabel("Algoritmo: ");
    speedLabel = new JLabel("Velocidad: ");
    orderTypeLabel = new JLabel("Orden: ");
    timeLabel = new JLabel("Tiempo: 00:00:000");
    stepsLabel = new JLabel("Pasos: 0");
    statsPanel.add(algorithmLabel);
    statsPanel.add(speedLabel);
    statsPanel.add(orderTypeLabel);
    statsPanel.add(timeLabel);
    statsPanel.add(stepsLabel);
    optionsPanel.add(statsPanel);

    add(optionsPanel, BorderLayout.NORTH);

    // Panel para la gráfica
    chartPanel = new ChartPanel(createChart());
    chartPanel.setPreferredSize(new Dimension(800, 450));
    add(chartPanel, BorderLayout.CENTER);

    // Evento del botón iniciar
    startButton.addActionListener(e -> startSorting());
}
```


MÉTODO: createChart()

Genera un gráfico de barras usando los datos actuales de `sortingData` y las etiquetas definidas.

```
private JFreeChart createChart() {
    DefaultCategoryDataset dataset = new DefaultCategoryDataset();
    for (DataEntry entry : sortingData) {
        dataset.addValue(entry.getCount(), yLabel, entry.getCategory());
    }
    return ChartFactory.createBarChart(
        "Proceso de Ordenamiento",
        xLabel,
        yLabel,
        dataset
    );
}
```

CLASE: Account

MainFrame es la ventana principal de la aplicación "USAC Processing Data". Se encarga de gestionar la interacción del usuario para cargar un archivo CSV, ingresar el título del gráfico y visualizar una gráfica de barras con los datos procesados. Además, ofrece la posibilidad de acceder al proceso de ordenamiento a través de un botón que abre la ventana modal SortDialog.

ATRIBUTOS Y VARIABLES

Método	Descripción
MainFrame()	Constructor que crea una instancia de MainController y llama a initUI() para configurar la interfaz gráfica.
initUI()	Configura la disposición de paneles, campos y botones, y asocia los eventos (carga, búsqueda y ordenamiento).
onBrowse()	Abre un cuadro de diálogo para seleccionar un archivo y muestra su ruta en el campo correspondiente.
onLoad()	Valida la entrada, carga los datos CSV mediante MainController, muestra el gráfico y habilita el botón de ordenamiento.
showChart(List<DataEntry>, String)	Genera y muestra un gráfico de barras con los datos cargados, usando las etiquetas de los ejes y el título ingresado.
onSort()	Abre la ventana modal SortDialog para iniciar el proceso de ordenamiento de los datos cargados.

MÉTODO: initUI()

Organiza la disposición de paneles, campos y botones, y asigna eventos a cada componente de la interfaz.

```
private void initUI() {
    setSize(1000, 700);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout());

    // Panel superior de entrada
    JPanel inputPanel = new JPanel();
    inputPanel.setLayout(new GridLayout(3, 1, 5, 5));
    inputPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

    // Panel de archivo
    JPanel filePanel = new JPanel(new BorderLayout(5, 5));
    filePathField = new JTextField();
    browseButton = new JButton("Buscar");
    filePanel.add(new JLabel("Ruta del archivo:"), BorderLayout.NORTH);
    filePanel.add(filePathField, BorderLayout.CENTER);
    filePanel.add(browseButton, BorderLayout.EAST);

    // Panel de título
    JPanel titlePanel = new JPanel(new BorderLayout(5, 5));
    titleField = new JTextField();
    titlePanel.add(new JLabel("Título para la gráfica:"),
        BorderLayout.NORTH);
    titlePanel.add(titleField, BorderLayout.CENTER);

    // Botón Aceptar
    JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
    loadButton = new JButton("Aceptar");
    buttonPanel.add(loadButton);

    // Añadir paneles al panel de entrada
    inputPanel.add(filePanel);
    inputPanel.add(titlePanel);
    inputPanel.add(buttonPanel);

    add(inputPanel, BorderLayout.NORTH);

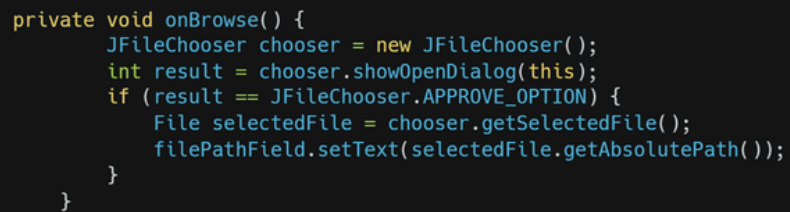
    // Contenedor de la gráfica
    chartPanelContainer = new JPanel(new BorderLayout());
    add(chartPanelContainer, BorderLayout.CENTER);

    // Botón para ordenar (inicialmente deshabilitado)
    sortButton = new JButton("Ordenar");
    sortButton.setEnabled(false);
    JPanel southPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));
    southPanel.add(sortButton);
    add(southPanel, BorderLayout.SOUTH);

    // Eventos
    browseButton.addActionListener(e -> onBrowse());
    loadButton.addActionListener(e -> onLoad());
    sortButton.addActionListener(e -> onSort());
}
```

MÉTODO: onBrowse()

Abre un diálogo para seleccionar un archivo CSV y muestra la ruta seleccionada en el campo correspondiente.



```
private void onBrowse() {  
    JFileChooser chooser = new JFileChooser();  
    int result = chooser.showOpenDialog(this);  
    if (result == JFileChooser.APPROVE_OPTION) {  
        File selectedFile = chooser.getSelectedFile();  
        filePathField.setText(selectedFile.getAbsolutePath());  
    }  
}
```